14195

Conversion of the Vehicle Survivability Analysis Tool (VSAT) from Unix to Window

James J. Crile, Roger W. Evans, John G. Bennett, and Jack Jones
U.S. Army Tank-Automotive Research, Development and Engineering Center
Warren, MI 48317-9000

ABSTRACT

Developed by Dynetics under an Army contract, the Vehicle Survivability Analysis Tool (VSAT) predicts the detectibility of ground vehicles to smart weapons that use radar and infrared sensors. Vehicle designers use VSAT to specify signature levels required to defeat threat sensors. Dynetics wrote VSAT to run under the Unix operating system on Silicon Graphics computers, the prevalent platform for signature modeling in the mid 1990's. Currently, however, Microsoft Windows computers perform much of the modeling previously possible only on Unix machines. After a brief introduction to VSAT itself, this paper describes our conversion of the program into a Windows version for modern computers. Problems to be overcome in the conversion included changes in function names between the Unix and Windows programming languages, packaging of FORTRAN code and new techniques for handling variables. Moreover, a Windows GUI was created to interface with the user. After the conversion, test cases run on the converted program verified that the Windows and Unix versions produced identical results.

INTRODUCTION

Dynetics developed the Vehicle Survivability Analysis Tool (VSAT) to meet an Army need for a methodology to define signature specifications for ground vehicles. Designers of vehicles needed to know how low a signature would have to be to defeat any given smart weapon system.

As a first step in analyzing these smart weapon systems, Dynetics classified them based on engagement technique and sensor technology. Weapons that engage targets while moving parallel to the ground are called fliers. And weapons that engage targets while descending are fallers. The sensors can be radar, infrared or dual mode. For each of these smart weapon systems, Dynetics created algorithms to model the engagement geometry and the detection process.

Using the appropriate algorithm and sensor parameters, VSAT can create a plot of signature level versus sensor performance for a given clutter level. For example, the detection probability versus radar cross section can be plotted for a weapon using a radar sensor, Figure 1. From this plot, a vehicle designer can write a signature specification to defeat the sensor.

As requested by the Army, Dynetics wrote VSAT to run on Silicon Graphics Incorporated (SGI) Unix computers because most signature modeling in the early 1990's was performed on such machines. Now, however, the Microsoft Windows operating system predominates. Therefore, the task described in this paper was undertaken to transport VSAT from Unix to Windows.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **29 JUL 2004**	2. REPORT TYPE **journal article**	3. DATES COVERED **29-07-2004 to 29-07-2004**

4. TITLE AND SUBTITLE **CONVERSION OF THE VEHICLE SURVIVABILITY ANALYSIS TOOL (VSAT) FROM UNIX TO WINDOW**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **James Crile; Roger Evans; John Bennett; Jack Jones**	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Army TARDEC ,6501 E.11 Mile Rd,Warren,MI,48397-5000**	8. PERFORMING ORGANIZATION REPORT NUMBER **#14195**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **U.S. Army TARDEC, 6501 E.11 Mile Rd, Warren, MI, 48397-5000**	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **#14195**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Ground Target Modeling and Validation Conference 2004

14. ABSTRACT
Developed by Dynetics under an Anny contract, the Vehicle Survivability Analysis Tool (VSAT) predicts the detectibility of ground vehicles to smart weapons that use radar and infrared sensors. Vehicle designers use VSAT to specify signature levels required to defeat threat sensors. Dynetics wrote VSAT to run under the Unix operating system on Silicon Graphics computers, the prevalent platform for signature modeling in the mid 1990's. Currently, however, Microsoft Windows computers perform much of the modeling previously possible only on Unix machines. After a brief introduction to VSAT itself, this paper describes our conversion of the program into a Windows version for modem computers. Problems to be overcome in the conversion included changes in function names between the Unix and Windows programming languages,packaging of FORTRAN code and new techniques for handling variables. Moreover, a Windows GUI was created to interface with the user. After the conversion, test cases run on the converted program verified that the Windows and Unix versions produced identical results.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**18**	

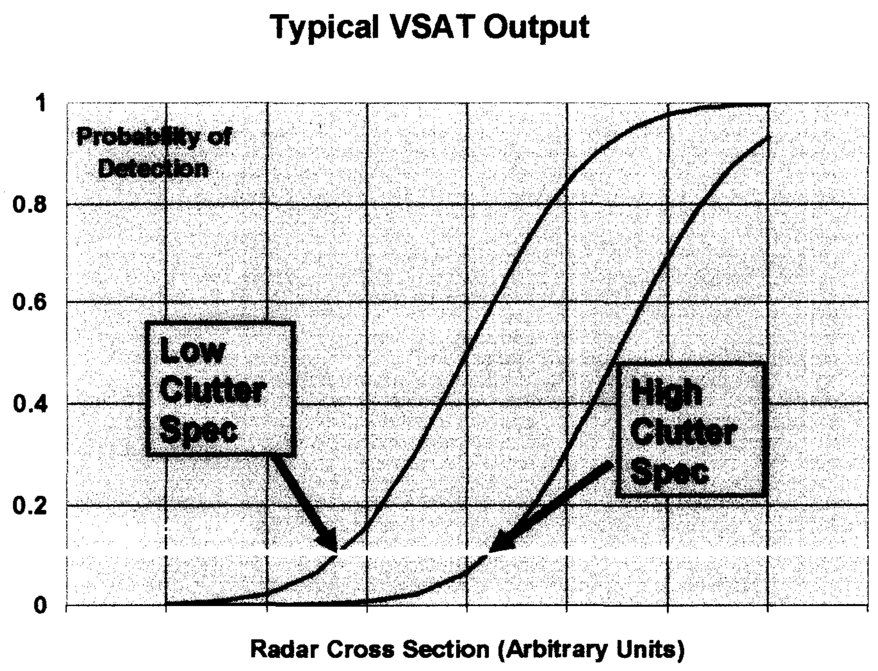


Figure 1. Plot of typical VSAT output

VSAT CONVERSION

The conversion of the VSAT code from the SGI to Windows platform consisted of converting two separate sets of code, one for the models and one for the graphical user interface (GUI). The model's C code largely remained untouched, but there were some troublesome issues with FORTRAN to be resolved. The GUI code was able to be reused to a large extent; but all of the graphical elements and event-driven functions had to be reworked considerably if not in totality. Moreover, the results of the model's program had to be verified and integrated with GUI code to finally complete the conversion of VSAT.

GUI Conversion

The conversion of the GUI took the most effort. After interviewing past users to see how they routinely used the program, we realized that VSAT contained some features that were rarely used. Moreover, some features were of limited or questionable usefulness, and others were prone to much user confusion and error. All these features were removed. Finally, instead of using unix-based GNU Plot for graphing results, it seemed much simple in our age of Microsoft Office to let users export the results as text and make better-looking graphs in Microsoft Excel. Therefore, what remains of VSAT is essentially a Windows version that would re-introduce it to the defense community at large while keeping the essentials of what made VSAT useful when based on the SGI.

The Process

Since the GUI-generator used to make the original VSAT was SGI-based and also no longer available, we recreated the GUI using Borland C++ Builder 5.0. The GUI was recreated in C++ while constantly comparing it to the SGI running on another machine. Print statements were placed in at least one place in every single function of the SGI version of VSAT. These statements would usually output the function name, file, and directory of the current function being run. Although this took some time, we were able to trace the execution of the code in an easy manner once the conversion began. This method of converting VSAT to a Windows-based machine proved fruitful. Although putting all the print statements in the SGI took quite a bit of time, this made debugging the Windows version much less complicated than it might have been otherwise.

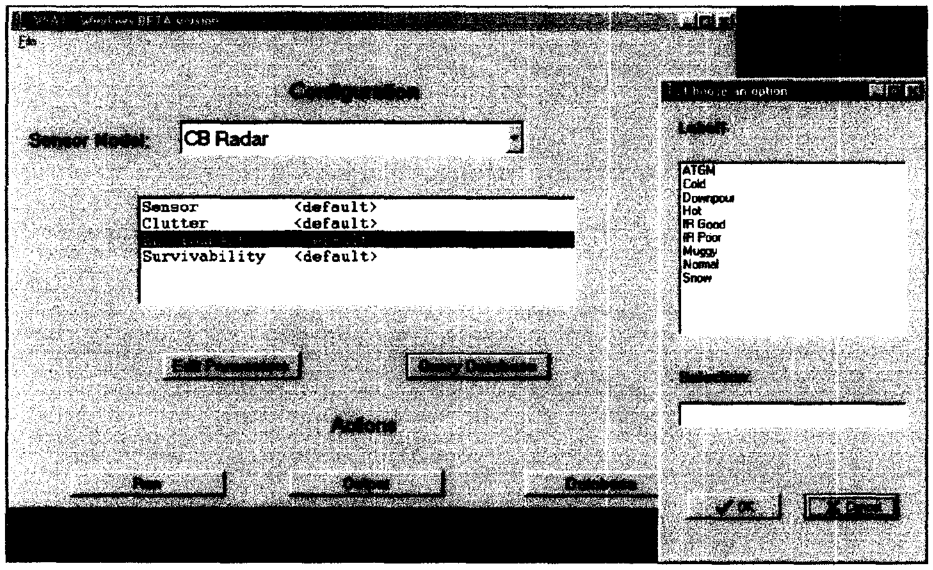
The conversion process was done in a top-down iterative manner as follows, incorporating a new feature through each iteration:

1.) Identify the next event-driven step to be converted.
2.) Locate the primary event driven function responsible for this event.
3.) Trace this main function through many of its statements, identifying functions called from the main function that also needed to be ported.
4.) Port or write necessary code in C++.
5.) Test.
6.) Debug (if necessary).

For instance, in viewing Figure 2, we can see the program has many buttons that trigger certain events, which in turn trigger others. We chose the events to be converted based on the following factors: logical necessity (one event comes logically after another), complexity of conversion, and actual value of the conversion to the finished project.

Therefore, based only on the figure below, we give a rough idea of the order of conversion. After each conversion or fixed bug, a version of the program was saved. Therefore, if we happened to make a mistake or wish to test a tentative feature, we had the option of moving to a previous version.

The steps in the conversion process were as follows:
1.) The program had to show up on the screen. This actually involves quite a bit of database reading and so forth.
2.) Changing the Sensor Model drop-down combo box.
3.) Query database button. The result is the box with the title 'Choose an option,' Logically, the next step here is to process what happens when a user clicks 'OK' or 'Cancel.
4.) Database button – viewing only.
5.) Edit Parameters. Since this event was much more complex than 2-4, it was fifth.
6.) Opening/Saving files from the file menu.
7.) Integrating GUI with models
8.) Output menu, exporting, and other finishing touches that could not be truly completed before the integration.



Sensor Model: CB Radar

```
Sensor          <default>
Clutter         <default>
Survivability   <default>
```

Figure 2. Display of VSAT with event-triggered dialog box

Conversion of FORTRAN Code

In addition to converting the GUI, some of the model code had to be converted from FORTRAN. The code that calculates VSAT uses the LOWTRAN (version 4.2, 1992) and NMMW (version unknown) models for calculating atmospheric transmittance. Both models are written in the FORTRAN programming language. Although LOWTRAN has been superseded by MODTRAN in recent times, it still works well for calculations not requiring a high spectral resolution.

On Unix systems, the C/C++ and FORTRAN compilers are usually provided as standard, and have been developed by the same company. This means that object code from each language can be linked together fairly easily. For Windows development, the easiest route is to create a DLL (dynamic link library) for the FORTRAN code which is then called by the C or C++ code. This was the approach taken for interfacing to the LOWTRAN and NMMW models.

The once-commercial Watcom FORTRAN product is now available as an open source package called OpenWatcom FORTRAN. This FORTRAN77 compiler is well suited to compiling legacy codes such as LOWTRAN and NMMW with minimal modification. The compiler and documentation are available from www.openwatcom.com. This compiler is capable of creating both Windows executables and DLLs.

All calculation variables are passed from C/C++ to FORTRAN in memory, avoiding all the possible problems which writing and reading files would introduce. For LOWTRAN, a standard parameter passing convention was used. For NMMW there were a larger number of variables being passed, so a FORTRAN common block was used for passing the variables both in and out. The LOWTRAN DLL we created is not general-purpose, since many input values have been hardcoded for VSAT use, although a more generic LOWTRAN module could easily be created should the need arise.

This method of using OpenWatcom FORTRAN to create a DLL file callable by modern C/C++ compilers works well, and requires much less effort than doing a total rewrite of the code. A report is available from the authors detailing the technique of creating a FORTRAN DLL and then calling it from C/C++.

Integrating of Models and GUI

The integration of the GUI with the models programs was relatively simple. The GUI program simply gives the cue to the command-line based models program to run. A DOS-Prompt appears and one can see that the program is executing from the messages rapidly moving down the screen. Quite simply, once the window disappears, we know the run has finished, and we can view the output. Communication between the programs does not exist because the development occurred on a Windows 98 machine.

VSAT Verification

Verification consisted of running a large number of test cases with both the SGI and Windows versions. The SGI version of VSAT had a script used to verify that the results of the models portion gave the correct results. This script executed as many of the branches of the code as possible. Although we were unable to use it for Windows version, we ran all of the approximately 50 cases manually and checked the resulting output files for accuracy, modifying the code if necessary. One case had an accuracy of 97.79%, two were between 98% and 99%, and the rest were over 99% accurate. Because of certain algorithms in the models, the variations were to be expected, and the verification was judged a success.

CONCLUSION

VSAT remains a useful tool for designers of combat vehicles. And the new Windows version of VSAT will make this tool available on modern computers.

Conversion of the Vehicle Survivability Analysis Tool (VSAT) from Unix to Window

James J. Crile, Roger W. Evans,
John G. Bennett, and Jack Jones
U.S. Army TARDEC

11 August 2004 GTMV Houghton, Michigan

Summary

- Introduction to VSAT
- Conversion from Unix to Windows
  - GUI
  - Sensor Models
- Verification
- Conclusions

Overview of VSAT

- Purpose: *To calculate the vehicle signature level required defeat a given sensor.*
- Sensors:
  - IR, Radar and Dual Mode Smart Munitions
  - Counter-Battery and Battlefield Radars

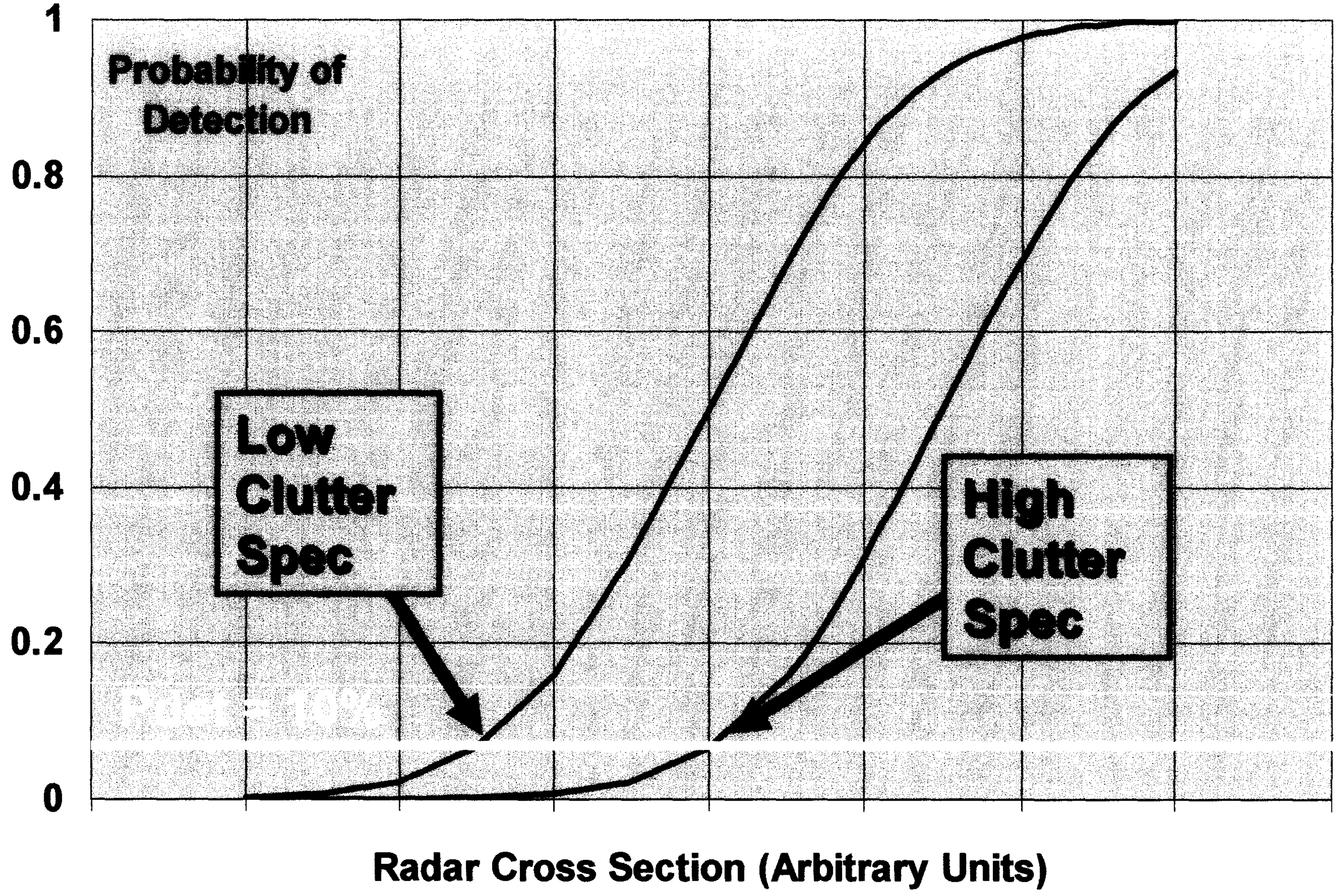
Inputs to VSAT

- Sensor Parameters
  - Selection from menu
  - Input of individual parameters
- Vehicle Overall Dimensions
- Environment
- Clutter Level
- Inputs Single Values or Range of Values

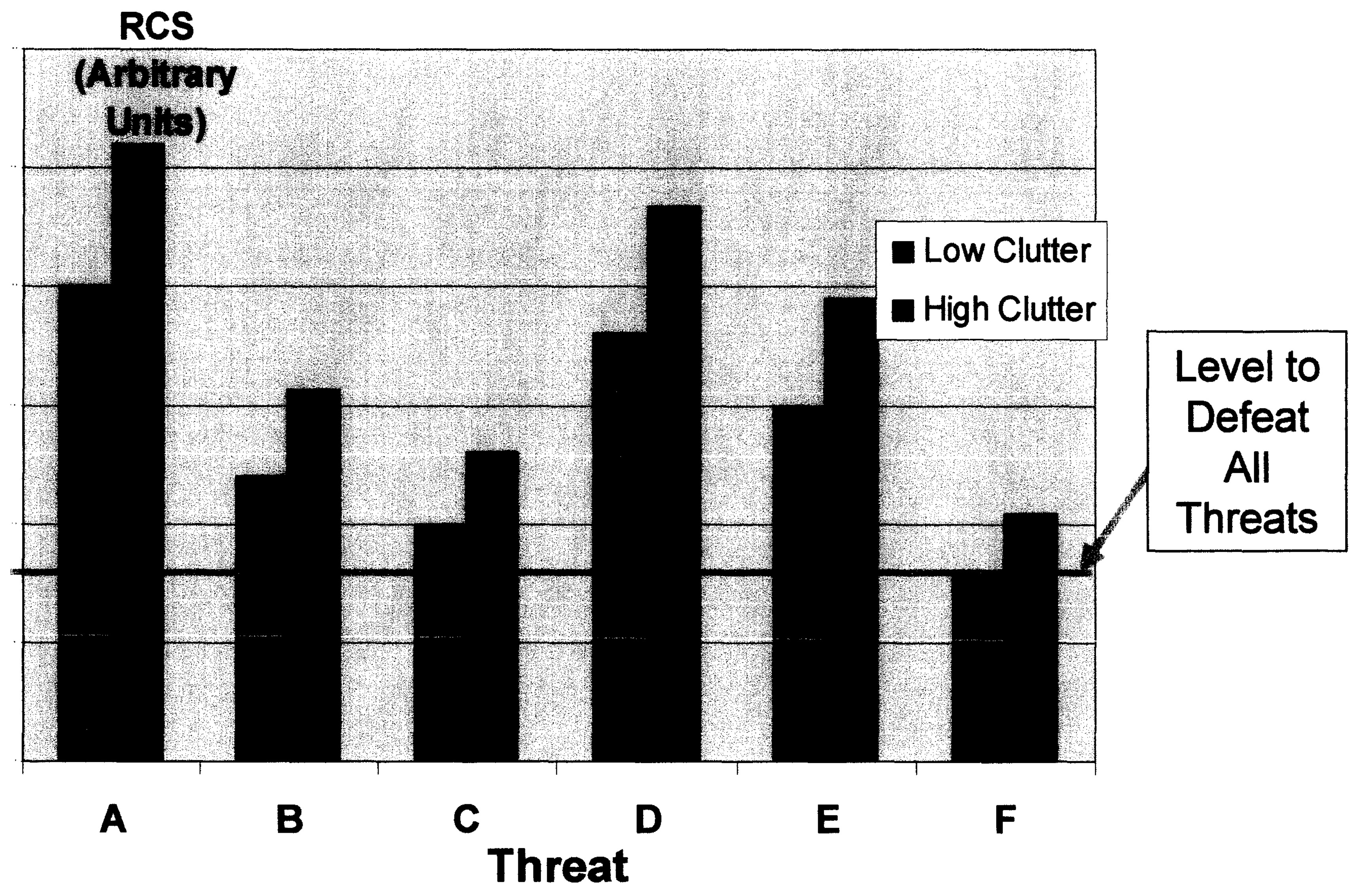
Outputs from VSAT

- Probability of Detection as a Function of
  - Clutter Level
  - Vehicle Signature
  - Sensor Parameters
- Required Signature Level for All Sensors

Typical VSAT Output



Required Signature vs. Threats



Conversion of VSAT

- VSAT originally written to run under SGI Unix
- Now need version to run under Microsoft Windows
- Conversion Tasks:
  - Convert GUI
  - Convert sensor models
  - Verify conversion

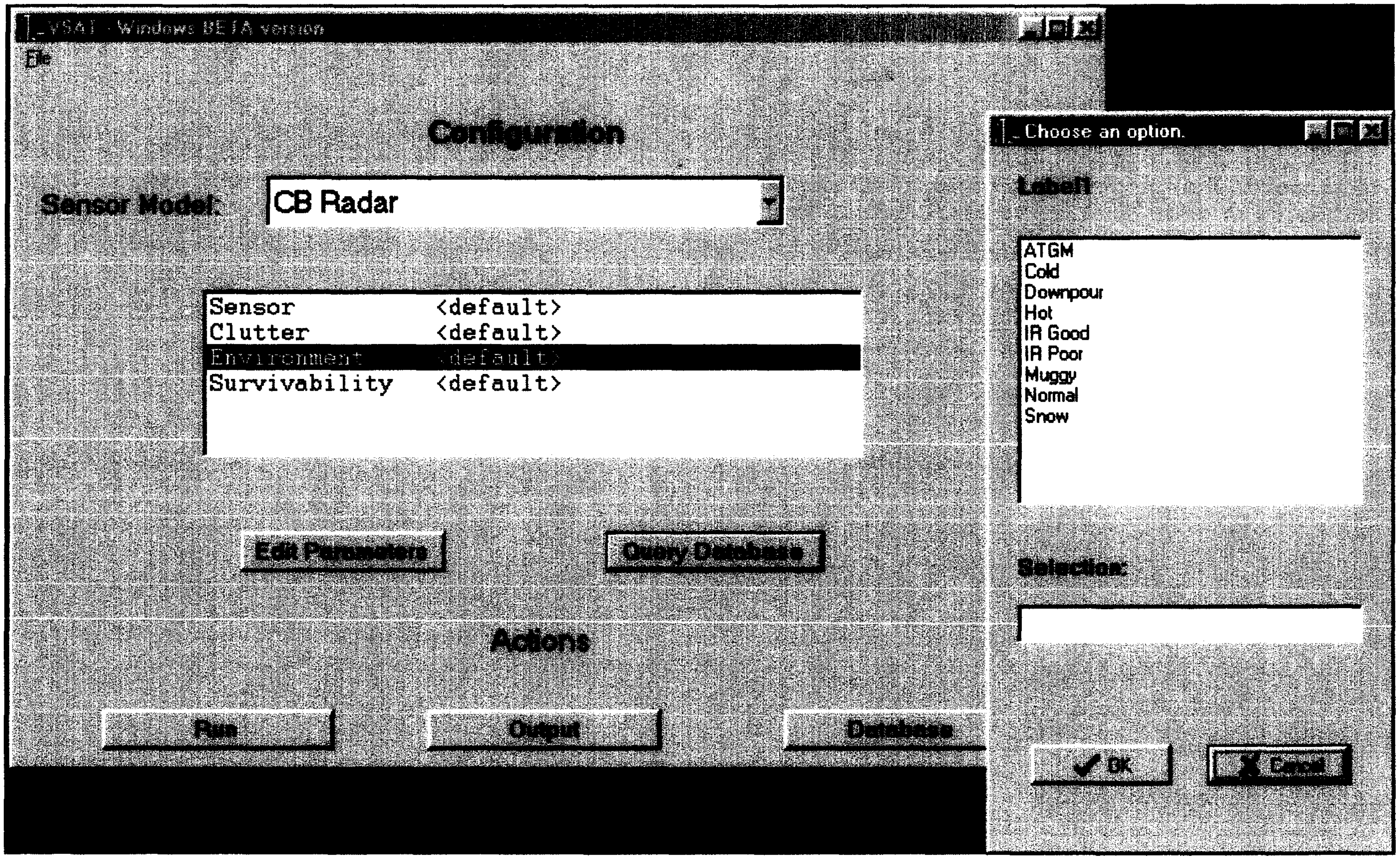
Conversion of GUI

- Rewrote GUI in Borland C++ Builder 5.0
- Placed Print statements in SGI code to track operation of code
- Wrote C++ code on an event-by-event basis

Conversion Process fro GUI

1. Identify the next event-driven step to be converted.
2. Locate the primary event driven function responsible for this event.
3. Trace this main function through many of its statements, identifying functions called from the main function that also needed to be ported.
4. Port or write necessary code in C++.
5. Test.
6. Debug as needed.

Conversion of GUI



Sensor Model: CB Radar

```
Sensor          <default>
Clutter         <default>
Environment     <default>
Survivability   <default>
```

ATGM
Cold
Downpour
Hot
IR Good
IR Poor
Muggy
Normal
Snow

Conversion of Sensor Models

- Original C code transferred directly to C++
- Atmospheric transmission models written in FORTRAN:
  - NMMW
  - LOWTRAN (version 4.2, 1992)
- For Windows, created a Dynamic Link Library (DDL) for FORTRAN code using OpenWatcom FORTRAN

Verification

- Compared runs from SGI and Windows versions
- Runs based on test script with original VSAT with 50 cases
- Match of cases was better than 99% for 47 cases, better than 98% for 2 cases and better than 97% for 1 case

Conclusions

- VSAT remains a useful tool for designers of combat vehicles.
- The new Windows version of VSAT will make this tool available on modern computers.